Preprint
UCRL-JC-144473

Performance of Neural Networks Methods in Intrusion Detection

V. N. P. Dao, R. Vemuri

This article was submitted to Cyber Defense Initiative, Washington, D. C., November 27 - December 3, 2001

July 9, 2001

U.S. Department of Energy

Lawrence
Livermore
National
Laboratory

This work was performed under the auspices of the United States Department of Energy by the University of California, Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

Performance of Neural Networks Methods in Intrusion Detection

Vu N.P. Dao [1]
dao1@llnl.gov

Rao Vemuri [1, 2]
rvemuri@ucdavis.edu

[1] Lawrence Livermore National Laboratory, 7000 East Ave., Livermore, CA 94551
[2] Department of Applied Science, University of California, Davis, CA 95616

Abstract

By accurately profiling the users via their unique attributes, it is possible to view the intrusion detection problem as a classification of authorized users and intruders. This paper demonstrates that artificial neural network (ANN) techniques can be used to solve this classification problem. Furthermore, the paper compares the performance of three neural networks methods in classifying authorized users and intruders using synthetically generated data. The three methods are the gradient descent back propagation (BP) with momentum, the conjugate gradient BP, and the quasi-Newton BP.

I. Introduction

Computer security issues are becoming headline grabbing items. According to one report, computer systems on the Internet are being attacked hundreds of times – each day [1]. One way of preventing computer attack is to detect intruders and stop them from accessing the protected computer networks. This paper focuses on intrusion detection and explores the potential of artificial neural networks as on-line, real-time tools for detecting intrusive activities. Toward this goal, the objective is to construct a model that captures a set of user attributes and determine if that set of user attributes belongs to the authorized users or to that of intruders.

The relevance of ANN's in intrusion detection becomes apparent when one views the intrusion detection problem as a pattern classification problem. Using profiles of authorized computer users [2], one can train an ANN to recognize the authorized users in the incoming traffic, thus separating authorized traffic from intrusion traffic.

More importantly, many of the current intrusion detection techniques cannot detect new and novel attack patterns. It is precisely in this area that the learning and generalization capabilities of ANN's can be exploited.

The rest of the paper is organized as follows. Section II formulates the problem and gives an overview of the different neural network methods used in detecting the intruders. Section III covers the generation of the test data sets. Section IV summarizes the results and compares the performances of the various neural network methods. Section V summarizes the research findings. Section VI gives a brief conclusion of the research and points to new research areas. The appendix lists the results.

II. Problem Formulation and The Back Propagation Methods

Mathematically speaking, the problem can be stated as follows. Given a *training data* set *S*, the goal is to establish a mapping f from any given input vector $\boldsymbol{x}$ to an output class d.

$$S = \left\{ \left(\mathbf{x}_i, y_i \right) : \mathbf{x}_i \in R^p, d \right\}$$

Where $\boldsymbol{x}$ is the input vector, defining a user's characteristic attributes and d is a scalar binary output to distinguish an authorized user from an intruder. From a modeling perspective, the objective is to seek a model that provides the best fit to the given training data and the best prediction capability with future observed data (also known as *test data* set), while minimizing model complexity. The above objective is typically accomplished by

building a model (i. e., the neural net, or equivalently, the mapping function f) and train it prior to using that model for intrusion detection. During the training phase, a performance index, defined in terms of the error, e, ($e = d - y$) is minimized. Where d is the desired output and y is the actual output.

The back propagation method is a popular technique to train multi-layer feed-forward neural networks in a supervised manner. This method is well known and details can be found in the published literature [3,4]. What follows is a brief synopsis of the methods and formulas used in this study. In each of these methods, the term "training" refers to the systematic procedure used to adjust the "weights" in a weighted sum of inputs that is responsible for activating a neuron. The sigmoidal activation [4] is used in this study.

In the gradient descent with momentum (GDM) [5,6], the new weight vector $\mathbf{w}_{k+1}$ is adjusted as:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha \ \mathbf{g}_k + \mu \ \mathbf{w}_{k-1}$$

Where α is the learning rate, $\mathbf{g}_k$ is the gradient of the error with respect to the weight vector, and μ is the momentum constant which can be any number between *0* and *1*.

In the conjugate gradient algorithms a search is performed along conjugate directions, which produces generally faster convergence than the steepest descent method. In the conjugate gradient algorithms the step size is adjusted at each iteration. A search is made along the conjugate gradient direction to determine the step size which will minimize the performance function along that line. The version of conjugate gradient method used here is due to Polak and Ribiere (CGP) [5,6]. The search at each iteration is determined by updating the weight vector as:

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \alpha \mathbf{p}_k$$

$$\text{where : } \mathbf{p}_k = -\mathbf{g}_k + \beta_k \mathbf{p}_{k-1}$$

$$\beta_k = \frac{\Delta \mathbf{g}_{k-1}^T \mathbf{g}_k}{\mathbf{g}_{k-1}^T \mathbf{g}_{k-1}}$$

$$\text{and} \qquad \Delta \mathbf{g}_{k-1}^T = \mathbf{g}_k^T - \mathbf{g}_{k-1}^T$$

Newton's method is an alternative to the conjugate gradient methods for fast optimization. Newton's method often converges faster than conjugate gradient methods. The weight update for the Newton's method is:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \mathbf{A}_k^{-1} \mathbf{g}_k$$

$\boldsymbol{A}_k$ is the Hessian matrix of the performance index at the current values of the weights and biases. When $\boldsymbol{A}_k$ is large, it is complex and time consuming to compute $\mathbf{w}_{k+1}$. Fortunately, there is a class of algorithms based on the works of Broyden, Fletcher, Goldfarb, and Shanno (BFGS) [4,7] that are based on Newton's method but which don't require intensive calculation. This new class of method is called quasi-Newton method. The new weight $\mathbf{w}_{k+1}$ is computed as a function of the gradient and the current weight $\mathbf{w}_k$.

III. Creation of Training and Test Data Sets

Creation of training data is central to the success of any neural networks method. One of the challenges is in creating intrusion scenarios. Earlier works have shown that computer users exhibited unique characteristics, and that computer users can be profiled accurately based on their attributes [2,8]. One way of collecting the data logs is to turn on the process accounting on UNIX operating system hosts and direct the output to a logfile. Other ways of collecting data logs can be from applications such as TCPdump, Snort, Syslog, and Dragon IDS [9]. The information collected in these logfiles ranges from the command type, login host, login time, CPU and memory usage to TCP header information such as the interface, source and destination addresses and ports, sequence numbers, window size, etc.

By turning on the UNIX OS process accounting, data logs were collected at the Computer Security Laboratory at the University of California, Davis. From these data logs, computer users were profiled based on four attributes [2]. These attributes were: (i) the command set used by the user, (ii) the login host, (iii) the time of login, and (iv) the time required to execute each of the commands entered (i.e. the CPU time). Two data sets for training and testing the neural networks were created. For simplicity in testing, these two data sets were created on the same host. Each of

these two data sets now has three attributes: command set, time of login and CPU time. The characteristic of the data set and how they were used for testing are covered next.

A. Characteristics of Data Set

The objectives of creating data sets for training and testing were twofold. First is to test the capability of the neural networks to classify data from authorized users and intruders that are linearly separated. Second is to test the performance in situations where the profiles of authorized users and intruders have some similarity. Two data sets were created based on four features above. In the first data set, the authorized users and intruders were separable from each other. The features in the second data were selected to blur the distinction between the authorized users and intruders. The characteristics of the two data sets are summarized in Table 1.

	User Features	Authorized Users	Intruders
First Data Set	Command Set	{1,3,6,7,8,11,14,17,19,21,22,25, 26, 28,29,31,33,37,38,41,43,44,46,47, 49,50,51,54,55,59,61,70,72,76,78, 79,81,84,85,87,91,92, 93,95,98,99}	{1,..,100}
	Login Host	1	1
	Login Time	6:00 – 19:00	1:00 – 5:00; 20:00 – 24:00
	CPU Time [μsec]	{1,2,3,4,5,6,7,8}	{50,60,70,80,90,100,200,300}
Second Data Set	Command Set	{1,3,6,7,8,11,14,17,19,21, 22,25,26, 28,29,31,33,37,38,41,43,44,46,47, 49,50,51,54,55,59,61,70,72,76,78, 79,81,84,85,87,91,92, 93,95,98,99}	{1,..,100}
	Login Host	1	1
	Login Time	6:00 – 19:00	1:00 – 7:00; 18:00 – 24:00
	CPU Time [μsec]	{1,2,3,4,5,6,7,8}	{5,6,7,8,10,20,40,50,60,70, 80,90,100,200, 300}

Table 1: User Profile Characteristics of Generated Test Data

B. Organization of Data Set

The next step is to organize the data set into a suitable format for training and testing. The generated data was organized into two parts. The first part is for training. Each training input sample came with a desired output. Here, ninety percent (90%) of the input data was generated as authorized traffic and ten percent (10%) as intrusion traffic. The second part is for testing the performance of the methods. In this part, ninety eight percent (98%) of the traffic were generated to be authorized traffic and two percent (2%) of the traffic to be intrusion traffic. In both parts of the training and test data section, several bursts of intrusion data are inserted into the authorized data stream. Each of the generated input data file has *7000* samples. The first *5000* samples are the training data, and the next *2000* samples are the test data. Authorized traffic is designated *Class Positive* and unauthorized traffic as *Class Negative.* Figure 1 illustrates the structure of the generated data files. Here an input sample is defined as a unit of data being fed into the neural network at one time; one sample can consist of many command units (CUs). A CU is defined as a set of four elements – the UNIX command, the login host, the time of login, and the execution time of the command (i.e. CPU time).

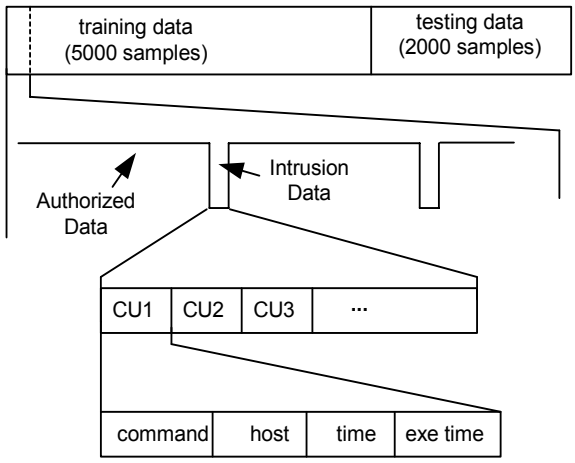

Figure 1: Construction of The Generated Data

The objective is to test the neural networks for detecting intrusion traffic with the fewest number of CUs. With that objective, three test files for the first data set and three test files for the second data set are generated. These six files are identified as follows: File1a, File1b, File2a, File2b, File3a and File3b. Each of the files File1a & File1b have 5 CUs in each input sample (that is, 5 x 4 = 20 input neurons). Thus File1a, for example, has 20 x 5,000 = 100,000 elements in the training section and 20 x 2000 = 40,000 elements in the test section. The same goes for File2a, File2b, File3a and File3b. The total number of elements for each input file is shown in Table 2.

	File1a File1b	**File2a File2b**	**File3a File3B**
Train Data	100,000	120,000	140,000
Test Data	40,000	48,000	56,000
Total	140,000	168,000	196,000

Table 2: Number of Elements in the Test Files

IV. Simulation Results and Performance Comparison

Simulation results are summarized in table 3 and table 4. In these tables, topology specifies the neural network architecture. For example, a topology of {20,10,1} indicates 20 input neurons, 10 hidden neurons, and 1 output neuron. The parameters $\alpha, \mu,$ indicate the learning rate and the momentum constants of the gradient descent BP with momentum. The quantity e is the mean of the square error (MSE) of the difference between the actual output y and the desired output d. An epoch is one complete presentation of the training data. 'FalsePos' and 'FalseNeg' are the false positive and false negative rates in classifying users. A false positive is classifying an authorized user as an intruder; conversely, a false negative is classifying an intruder as an authorized user. Each method terminates when any of the following condition occurs: (i) MSE = exp(-5), (ii) Epoch = 500, and (iii) when the gradient undergoes negligible change from one epoch to the next (i.e. typically exp(-10) in the simulation).

Table 3 summarizes results obtained when the first data set was applied as input. Since the gradient descent with momentum (GDM) method did not perform as well as the other two methods (CGP and BFGS), it was excluded from being used to test the second data set. Table 4 summarizes results obtained when the second data set was used as input for the conjugate gradient and quasi Newton methods.

	File 1a Input		**File 2a Input**		**File 3a Input**	
GDM	Topology {20,17,1}	FalsePos = N/A	Topology {24,14,1}	FalsePos = 0	Topology {28,17,1}	FalsePos = 0
	α = 0.1, μ = 0.75	False Neg =N/A	α = 0.05, μ = 0.75	FalseNeg = 0	α = 0.05, μ = 0.70	FalseNeg = 75%
	MSE = N/A	500 Epoch	MSE = exp-5	140 Epoch	MSE = 0.07	500 Epoch
CGP	Topology {20,13,1}	FalsePos = 0	Topology {24,10,1}	FalsePos = 0	Topology {28,13,1}	FalsePos = 0
		FalseNeg = 0		FalseNeg = 0		FalseNeg = 0
	MSE = exp-5	90 Epoch	MSE = exp-5	50 Epoch	MSE = exp-5	40 Epoch
BFGS	Topology {20,13,1}	FalsePos = 0	Topology {24,11,1}	FalsePos = 0	Topology {28,11,1}	FalsePos = 0
		FalseNeg = 0		FalseNeg = 0		FalseNeg = 0
	MSE = exp-5	45 Epoch	MSE = exp-5	60 Epoch	MSE = exp-5	40 Epoch

Table 3: Summary of Simulation Results for the First Data Set

	File 1b Input		File 2b Input		File 3b Input	
CGP	Topology {20,15,1}	FalsePos = .04% FalseNeg = 15%	Topology {24,14,1}	FalsePos = 0.01% FalseNeg = 17.5%	Topology {28,16,1}	FalsePos = 0 FalseNeg=17.5%
	MSE = 0.01	500 Epoch	MSE = 0.17	500 Epoch	MSE = 0.18	450 Epoch*
BFGS	Topology {20,17,1}	FalsePos=0.15% FalseNeg=12.5%	Topology {24,15,1}	FalsePos=0.15% FalseNeg = 5.0%	Topology {28,16,1}	FalsePos= 0.05% FalseNeg=25.0%
	MSE = 0.01	500 Epoch	MSE = 0.01	500 Epoch	MSE	100 Epoch*

Table 4: Summary of Simulation Results for the Second Data Set
*NN method stops due to gradient too small

In testing the following observations were made. First, the gradient descent with momentum method did not perform well as the Conjugate Gradient and quasi Newton methods. Furthermore, it was time consuming to tune the learning rate, α, and the momentum constant, μ, for this method to operate properly. For instance, when the input was File1a, the GDM method was not able to classify the intrusion traffic from the authorized traffic, however it was able to do the classification correctly when the input file was File2a or File3a. When the input file was File2a, GDM requires 140 epochs to converge compared to only 50 epochs for CGP and 60 epochs for BFGS. When the input file was File3a, GDM needed 500 epochs to converge to a 75% false negative error compared to 40 epochs and no false negative for both CGP and BFGS methods. These results indicate that both the CGP and the BFGS methods were superior in classifying authorized users from intruders while maintaining good false negative values and good convergence properties compared to the GDM method.

Second, the conjugate gradient and the quasi Newton methods exhibited roughly the same performance in terms of classification. Both of these methods required simpler NN topology (i.e. NN topology with fewer hidden layer neurons) and fewer epochs to converge compared to the GDM method.

Third, the number of CUs used in each input sample affected the performance of the classification of the data. Of all the data sets used, File2a and File2b tended to yield the best performance for the three neural networks methods. This leads one to believe that an input sample consisting of 5 CUs (20 elements) might not contain enough information, while 7 CUs (28 elements) input sample might contain too much information, whereas 6 CUs (24 elements) input samples contains the right amount of information for classification of computer users. This issue needs further study.

V. Summary

In this paper, the problem of intrusion detection is posed as a classification problem. Three different BP neural network methods were applied to solve this problem. For training and testing purposes, two synthetic test data sets were created. From the first test data set, it was shown that the three neural networks methods were capable of classifying authorized users from intruders. The conjugate gradient and the quasi Newton methods yielded superior performance than the gradient descent with momentum method. With the second test data set, the conjugate gradient and quasi Newton methods performed best when each input sample size is 6 CUs, or equivalently 24 elements long.

VI. Conclusion and Future Work

From the preliminary results shown in this paper it appears that neural network techniques hold some potential in intrusion detection. The training data and test data used in this study were synthetically generated. Using this generated data, it was shown that the conjugate gradient and the quasi Newton can successfully detect intruders logging into a computer network. These two methods only required an input sample of 6

consecutive CUs from the intruder to classify that the current user is indeed an intruder.

Several issues remain to be investigated. First, it is necessary to evaluate the performance of neural networks with data sets captured by monitoring real traffic. Second, it is also necessary to establish the feature set that best describes users and intruders. The three used in this paper, namely command set, login time, and CPU usage, were selected because they appeared to be reasonable choices after a cursory examination of a data set available at the Security Laboratory of the University of California at Davis. Third, it is necessary to characterize the drifting patterns among authorized users and develop an incremental training procedure. Finally, besides the BP methods, there are other neural networks methods like the radial basis function (RBF) that could be implemented. These issues are being addressed at this time.

VII. Acknowledgement

This work was performed under the auspices of the U.S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under Contract No. W-7405-Eng-48.

VIII. References

[1] R. Bace, *Intrusion Detection*, Macmillan Technical Publishing, 2000.

[2] V. Dao, R. Vemuri, S. Templeton, "Profiling Users in the UNIX OS Environment", *International Computer Science Conventions Conference*, University of Wollongong, Australia, Dec. 2000.

[3] J. Principe, N. Euliano, W. Lefebvre, *Neural and Adaptive System – Fundamentals Through Simulations*, Wiley, 2000.

[4] S. Haykin, *Neural Networks – A Comprehensive Foundation*, 2nd Edition, Prentice Hall, 2000.

[5] MATLAB, *Neural Network Toolbox*, Version 3, The Math Works Inc., 1998.

[6] E. Dennis, R. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Englewood Cliffs, NJ: Prentice-Hall, 1983.

[7] M. Hagen, H. Demuth, and M. Beale, *Neural Network Design*, Boston, MA., PWS Publishing, 1996.

[8] D. Denning, "An Intrusion Detection Model", *IEEE Transactions on Software Engineering*, 1987.

[9] S. Northcutt, M. Cooper, M. Fearnow, K. Frederick, *Intrusion Signature and Analysis*, New Riders, 2001.